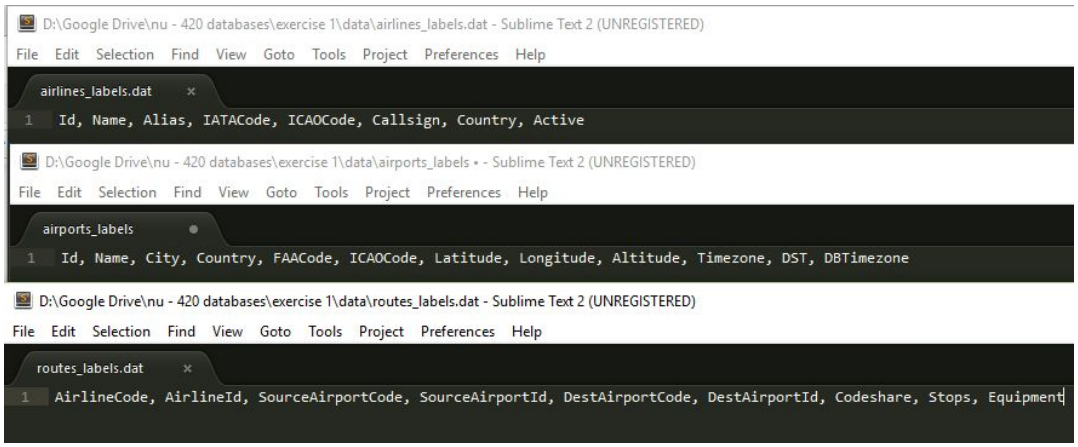
MSPA 420 - Graded Exercise 1

This submission includes the results of doing the full assignment (not just the microassignment). I used this assignment to learn about pandas, as well as making service oriented API calls, as well as how to create objects in Python.

I'm currently trying to figure out what my favorite Python tool is. I mainly am using Visual Studio's Python plugin, because I love the intellisense features. I'd prefer to find a tool that is more like R Studio if you can recommend something. It helps to have that working space where you can inspect objects more easily that the Python IDEs I've used so far. The main OS that I'm using is Windows 10 64bit.

The variables/labels used for airlines, airports, and routes:




```
import pandas as pd
import pickle
import os
import requests          # for making http request to google api
import geopy             # for geolocation distance
from geopy.distance import vincenty
import math
import numpy as np

# class to house source information from the files and act on the source
# information
class AirInfo:
    def __init__(self,airports, airlines, routes):
        self.airports = airports
        self.airlines = airlines
        self.routes = routes

    def saveWithHeaders(self, outputDirectoryPath):
        saveFileWithHeader(self.airports,outputDirectoryPath, "airports.p")
        saveFileWithHeader(self.airlines,outputDirectoryPath, "airlines.p")
        saveFileWithHeader(self.routes,outputDirectoryPath, "routes.p")
```

```
# class to house geolocation information about the user of the program
# also experimenting with object oriented programming
class MyGeoInfo(object):
    googlemapskey = "AIzaSyCzJ_8SbqxWuXFVKQK15wqRBAGriHkbJPE"
    _longitude = -1
    _latitude = -1
    _closestAirport = -1

    def __init__(self, city, state):
        # get the coordinates for the city and state passed in
        # decided to use the google api to get the coordinates
        response = requests.get("https://maps.googleapis.com/maps/api/geocode/json?address=" +
                                city + ",+" +
                                state +"&key=" +
                                MyGeoInfo.googlemapskey)
        resp_json_payload = response.json()
        self._longitude = resp_json_payload['results'][0]['geometry']['location']['lng']
        self._latitude = resp_json_payload['results'][0]['geometry']['location']['lat']

    @property
    def Longitude(self):
        return self._longitude
    @property
    def Latitude(self):
        return self._latitude
    @property
    def ClosestAirport(self):
        return  self._closestAirport
    @ClosestAirport.setter
    def ClosestAirport(self, id):
        self._closestAirport = id

    # get the distance between my location and another location
    def getDistance(self, otherLongitude, otherLatitude):
        myLocation = (self.Latitude, self.Longitude)
        otherLocation = (otherLatitude, otherLongitude)
        return geopy.distance.vincenty(myLocation, otherLocation).miles

# given a set of data and a directory and filename, this function
# will store the data into the file specified.
def saveFileWithHeader(data, outputDirectoryPath, fileName):
    data.to_pickle(os.path.join(outputDirectoryPath,fileName))

def loadFiles():
    # source file locations
    srcAirportsPath = r"D:\Google Drive\nu - 420 databases\exercise 1\data\airports.dat"
    srcAirlinesPath = r"D:\Google Drive\nu - 420 databases\exercise 1\data\airlines.dat"
    srcRoutesPath = r"D:\Google Drive\nu - 420 databases\exercise 1\data\routes.dat"
    srcAirportsLabelsPath = r"D:\Google Drive\nu - 420 databases\exercise 1\data\airports_labels.dat"
    srcAirlinesLabelsPath = r"D:\Google Drive\nu - 420 databases\exercise 1\data\airlines_labels.dat"
    srcRoutesLabelsPath = r"D:\Google Drive\nu - 420 databases\exercise 1\data\routes_labels.dat"

    # read in the files, handle encoding, types, and missing values
    airportLabels = pd.read_csv(srcAirportsLabelsPath, encoding='latin-1', skipinitialspace=True)
    airports = pd.read_csv(srcAirportsPath, names=airportLabels, encoding='latin-1',
                        na_values = ["N/A","\\N"],
                        dtype={'Id':float,'Name':'str'})
```

```
                    #, nrows=20)

    airlinesLabels = pd.read_csv(srcAirlinesLabelsPath, encoding='latin-1', skipinitialspace=True)
    airlines = pd.read_csv(srcAirlinesPath, names=airlinesLabels, encoding='latin-1',na_values = ["N/A","\\N"])

    routesLabels = pd.read_csv(srcRoutesLabelsPath, encoding='latin-1', skipinitialspace=True)
    routes = pd.read_csv(srcRoutesPath, names=routesLabels, encoding='latin-1',
                    na_values = ["N/A","\\N"],
                    dtype={'SourceAirportId':float,'DestAirportId':float})

    # return an object containing all the source information
    return(AirInfo(airports,airlines,routes))

def findClosestAirport(geo, airports):
    # set minimum distance to a high value so any airport would become closer
    minDistance = 1000000000000
    closestAirport = ''
    for row in airports.iterrows():
        airport = row[1]
        # some airports in the data don't have codes, so skip over those
        faaCode = airport['FAACode']
        if type(faaCode).__name__ != "unicode":
            continue
        icaoCode = airport['ICAOCode']
        if type(icaoCode).__name__ != "unicode":
            continue
        latitude = airport['Latitude']
        longitude = airport['Longitude']
        distance = geo.getDistance(longitude, latitude)
        if distance < minDistance:
            minDistance = distance
            closestAirport = airport['Id']
    return (closestAirport, minDistance)

##########################################################################################
# main processing
air = loadFiles()
air.saveWithHeaders("D:\\Google Drive\\nu - 420 databases\\exercise 1\\data\\")

# set distance information
me = MyGeoInfo('st louis','mo')
closestInfo = findClosestAirport(me,air.airports)
me.ClosestAirport = closestInfo[0]
me.DistanceToClosestAirport = round(closestInfo[1],2)

# print the closest airport
closestAirportName = air.airports[air.airports['Id'] == me.ClosestAirport]["Name"].values[0]
closestAirportCode = air.airports[air.airports['Id'] == me.ClosestAirport]["FAACode"].values[0]
print(closestAirportName + " (" + closestAirportCode + ")" + " is the closest airport at "
      + str(me.DistanceToClosestAirport) + " miles away")

# print the number of routes departing from the closest airport
departingRoutes = air.routes[air.routes['SourceAirportId'] == me.ClosestAirport]
print(str(len(departingRoutes)) + " flights departing from " + closestAirportCode)

# print the flights arriving into EGO
egoRoutes = air.routes[air.routes['DestAirportCode'] == "EGO"]
```

```
print(str(len(egoRoutes)) + " flights arriving into EGO")

k=raw_input('')
##########################################################################################
```

This is the output for saying I live in St. Louis. I programmed it using the Google geolocation API so that I can set any city or state and it'll compute the closest airport.

```
D:\Program Files\Python\Python27\python.exe
Lambert St Louis Intl (STL) is the closest airport at 12.47 miles away
114 flights departing from STL
11 flights arriving into EGO
```